iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79652

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Automated Handwriting Recognition and Digital Document Transformation Using Vision Transformers and TrOCR

Mohamed Haleem Akmal S[1], Rohan Karthik R S[2], Vasanthakumar S[3], Manojkumar P[4], Ms. R. Kavitha[5]
[1, 2, 3, 4]*B.Tech Information Technology, M.I.E.T Engineering College, Tiruchirappalli, Tamil Nadu, India*
[5]*Assistant Professor, Department of Information Technology, M.I.E.T Engineering College, Tiruchirappalli, Tamil Nadu, India*

Abstract: ***This paper presents an Automated Handwriting Recognition and Digital Document Transformation System de- signed to convert handwritten documents into editable digital formats. The system accepts scanned PDFs and handwritten images as input and performs image preprocessing and noise reduction to improve text clarity. A Vision Transformer (ViT) encoder is employed to extract visual features from image patches, while a Transformer-based decoder generates character sequences using self-attention and cross-attention mechanisms. The core recognition model utilises TrOCR for end-to-end hand- writing recognition without requiring character segmentation. Post-processing algorithms such as token decoding, spell correc- tion, and text formatting are applied to enhance readability and consistency. Finally, the recognised text is exported into digital formats such as TXT, DOCX, and searchable PDF, enabling efficient and accurate digital document transformation.***
Index Terms: Handwriting Recognition, TrOCR, Vision Transformer, OCR, Document Digitisation, Transformer Decoder, NLP, Image Preprocessing, Deep Learning.

I. INTRODUCTION

Handwritten documents remain one of the most natural and widely used forms of human communication. From historical manuscripts and medical prescriptions to student notes and government records, vast amounts of valuable information still exist only in handwritten form. Converting these documents into machine-readable, editable digital text is a longstanding challenge in computer vision and natural language processing. Traditional Optical Character Recognition (OCR) systems were designed primarily for printed text and rely heavily on character segmentation—isolating individual characters before recognising them. This pipeline breaks down severely when applied to cursive or free-form handwriting, where character boundaries are ambiguous and stroke styles vary dramatically between individuals. Noise in scanned documents, skewed images, and low-resolution inputs compound the problem further.

Recent advances in transformer-based architectures have fundamentally changed the landscape of sequence prediction tasks. Models such as BERT [1] and GPT [2] demonstrated that attention mechanisms can capture long-range dependen- cies in text with remarkable accuracy. The application of this paradigm to vision tasks, through Vision Transformers (ViT) [3], opened new possibilities for image understanding without relying on convolutional inductive biases. Building on these foundations, TrOCR [4] introduced an end-to-end trainable encoder-decoder architecture that treats handwriting recognition as a sequence-to-sequence problem, bypassing the need for explicit character segmentation entirely.

Our system, built as a final-year B.Tech project at M.I.E.T Engineering College, Tiruchirappalli, integrates TrOCR into a full document processing pipeline.

The contributions of this work are:

1) An end-to-end document ingestion pipeline supporting handwritten JPG/PNG images and multi-page scanned PDFs.
2) A preprocessing module applying grayscale conversion, noise removal, skew correction, and contrast enhancement to standardise inputs for the recognition model.
3) Integration of the TrOCR model (ViT encoder + Trans- former decoder) for segmentation-free, end-to-end hand- writing recognition.
4) A post-processing module performing token decoding, spell correction, multi-page merging, and final export to TXT, DOCX, and searchable PDF formats.
5) A Flask-based web interface enabling non-technical users to upload documents and download digitised outputs.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

II. RELATED WORK

A. *Traditional OCR Systems*

Early OCR systems relied on rule-based image process- ing: thresholding, connected-component analysis, and template matching. Tesseract [5], one of the most widely deployed open-source engines, performs well on clean printed text but degrades significantly on handwritten input. Its pipeline assumes character isolation, which fails for cursive script. Our system specifically targets this gap by replacing segmentation- based recognition with an attention-driven sequence model.

B. *Convolutional Approaches to Handwriting Recognition*

Convolutional Recurrent Neural Networks (CRNN) with Connectionist Temporal Classification (CTC) [6] advanced the state of the art by removing the need for explicit segmentation at the word level.

However, CTC-based models still struggle with long-range character dependencies and are sensitive to sequence length mismatches. Our approach replaces CTC entirely with a Transformer decoder, which handles variable- length outputs more naturally through autoregressive generation.

C. *Vision Transformers*

Dosovitskiy et al. [3] introduced the Vision Transformer, demonstrating that a pure attention-based architecture could match or exceed convolutional networks on image classi- fication benchmarks when pre-trained on sufficiently large datasets. By dividing an image into fixed-size patches and treating each patch as a token, ViT enables global context modelling from the first layer. We adopt the ViT encoder within TrOCR to extract patch-level visual features from preprocessed handwriting images.

D. *TrOCR*

Li et al. [4] proposed TrOCR, an encoder-decoder model pre-trained on large-scale synthetic and real handwriting datasets. The encoder is a ViT-Base or ViT-Large model; the decoder is initialised from a language model (e.g., RoBERTa or GPT-2). Cross-attention in the decoder attends to the encoder's patch representations while autoregressively predicting character tokens. TrOCR achieved state-of-the- art results on the IAM Handwriting Dataset [7] without any character-level segmentation. Our project fine-tunes and deploys the microsoft/trocr-base-handwritten checkpoint within a production-ready web application.

E. *Document Digitisation Pipelines*

Prior document processing systems such as ABBYY FineReader and Adobe Acrobat OCR handle printed docu- ments effectively but offer limited support for free-form hand- writing. Open-source alternatives like EasyOCR [8] extend coverage to multiple scripts but still rely on convolutional backbones. Our system is differentiated by its Transformer- only recognition core and its focus on producing multiple editable output formats with post-processing applied before export.

III. PROBLEM STATEMENT

Despite decades of research in OCR, handwriting recogni- tion in real-world deployments remains an unsolved problem for several practical reasons:

1) Segmentation dependency: Traditional systems require explicit character boundaries. Cursive and connected scripts violate these assumptions, causing cascading recognition errors.
2) Input quality variability: Scanned handwritten docu- ments are noisy, skewed, and inconsistently lit. These artefacts degrade feature extraction at every stage of a segmentation pipeline.
3) Lack of contextual understanding: Character-level models do not leverage word or sentence context. A misrecognised character cannot be corrected downstream without language-level knowledge.
4) Limited output formats: Most OCR tools produce plain text only. Users needing editable DOCX or searchable PDF must rely on separate post-processing tools, increas- ing friction significantly.
5) Multi-page handling: Long documents require page- level processing and coherent merging of recognised text across pages—a capability rarely integrated into a single pipeline.

Our system addresses each of these limitations through a unified, end-to-end pipeline that takes a raw uploaded file and returns a ready-to-use digital document.

IV. OBJECTIVES

The specific objectives of this project are:

1) To accept handwritten images (JPG/PNG) and scanned PDF documents as input through a web interface.
2) To preprocess and enhance document images for im- proved recognition accuracy using grayscale conversion, resizing, noise removal, skew correction, and contrast enhancement.
3) To extract visual features from preprocessed images using a Vision Transformer (ViT) encoder operating on fixed-size image patches.
4) To recognise handwritten text using the TrOCR model without requiring character-level segmentation, leverag- ing cross-attention between visual features and predicted token sequences.
5) To apply post-processing steps including token decod- ing, spell correction, and text formatting to produce clean, readable output.
6) To convert the recognised text into editable digital formats: plain TXT, Microsoft Word DOCX, and search- able PDF.
7) To reduce manual transcription effort and turnaround time for document digitisation in educational, healthcare, and government contexts.

V. SYSTEM ARCHITECTURE

The system is structured as a four-layer pipeline: a web- based frontend, a Flask API layer, a processing middleware block, and a model execution and output tier. Figure 1 illustrates the full architecture.

A. Frontend (Web Interface)

A Flask-rendered HTML/CSS/JavaScript interface allows users to upload handwritten images or scanned PDFs. The interface provides a file picker, a progress indicator during pro- cessing, and a download panel once digitisation is complete. No local software installation is required from the user's side.

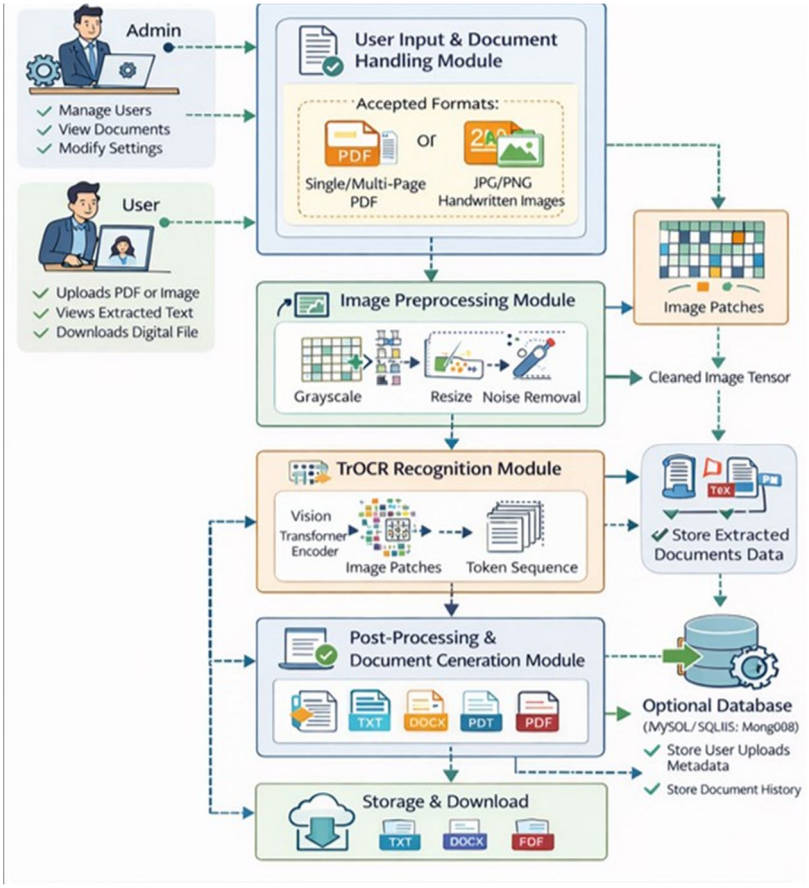


Fig. 1. System Architecture — end-to-end document transformation pipeline.

B. API and Routing Layer

The Flask backend exposes three primary endpoints: /upload for file submission, /status for polling process- ing progress, and /download for retrieving output files. Re- quest validation checks file type and size before any processing begins, returning structured error messages for unsupported inputs.

C. Preprocessing Module

Incoming files are routed through a type detector. PDFs are converted page-by-page into high-resolution PNG images using pdf2image. All images are then passed through the preprocessing pipeline: grayscale conversion, resizing to the model's expected input resolution (384×384 for TrOCR-base), pixel normalisation, noise removal via Gaussian filtering, skew correction using Hough transform-based angle detection, and contrast enhancement via adaptive histogram equalisation (CLAHE). The output is a standardised image tensor ready for the recognition model.

D. TrOCR Recognition Module

The preprocessed tensor is fed into the microsoft/trocr-base-handwritten model. The ViT encoder divides the image into non-overlapping 16×16 patches, projects each patch into an embedding, and processes the sequence through multi-head self-attention layers to produce patch-level feature representations. The Transformer decoder takes these encoder outputs and, using cross-attention and causal self-attention, autoregressively predicts a token sequence representing the recognised characters. Beam search with a beam width of 4 is used during inference to improve output quality over greedy decoding.

E. Post-Processing and Output Generation

The token sequence is decoded using the model's tokeniser. A spell-correction pass using pyspellchecker corrects isolated recognition errors that produce non-dictionary words. Multi-page documents have their per-page text blocks con- catenated with appropriate paragraph breaks. The cleaned text is then exported in three formats: plain TXT via Python's built-in file I/O, DOCX via the python-docx library, and searchable PDF via reportlab with embedded text layer over the original page images.

VI. MODULE DESCRIPTION

A. Module 1: User Input and Document Handling

1) Purpose: Accept handwritten documents and identify the document type for routing.

2) Input: Handwritten image (JPG/PNG) or scanned PDF (single or multi-page).

3) Process: The user uploads a file through the web interface. The system detects the file type using the MIME type and file extension. If the input is a PDF, each page is converted into a high-resolution image using pdf2image (which wraps Poppler). If the input is an image file, it is passed directly to the preprocessing module.

4) Output: One or more images ready for preprocessing, along with metadata (file name, page count, file type).

B. Module 2: Image Preprocessing

1) Purpose: Improve image quality and standardise inputs to maximise recognition accuracy.

2) Input: Raw image(s) from the input module.

3) Process:

- Convert to grayscale using the luminance-preserving formula Y = 0.299R + 0.587G + 0.114B.
- Resize to the model's target resolution (384×384 pixels) using bicubic interpolation.
- Normalise pixel values to the [0, 1] range and apply ImageNet mean/std normalisation.
- Remove noise using a Gaussian blur kernel with σ = 1.0 to suppress high-frequency artefacts while preserving stroke edges.
- Detect and correct document skew by computing the dominant line angle via the Hough Line Transform and rotating the image to align text horizontally.
- Enhance contrast using CLAHE with a clip limit of 2.0 and a tile grid of 8×8 to improve local contrast in low- illumination regions.

4) Output: Cleaned and standardised image tensor of shape (3, 384, 384).

C. *Module 3: TrOCR Recognition*

1) Purpose: Extract handwritten text from the preprocessed image using a Transformer-based OCR model.
2) Input: Preprocessed image tensor.
3) Process: The ViT encoder divides the input into 576 non- overlapping 16×16 patches. Each patch is linearly projectedto a 768-dimensional embedding and augmented with a learn- able positional embedding. The sequence passes through 12 Transformer encoder layers, each with 12 attention heads. The resulting patch-level context vectors are passed as key- value pairs to the Transformer decoder. The decoder applies causal self-attention over previously predicted tokens and cross-attention over encoder outputs to predict the next token autoregressively. Decoding terminates at the end-of-sequence token.
4) Output: Token ID sequence representing the recognised handwritten text.

D. *Module 4: Post-Processing and Document Generation*

1) Purpose: Refine extracted text and generate downloadable digital documents.
2) Input: Token sequence from the TrOCR model.
3) Process:

- Decode token IDs to Unicode text using the model's tokeniser with skip_special_tokens=True.
- Run spell correction using pyspellchecker to re-
- place out-of-vocabulary tokens with their most probable in-vocabulary correction.
- Merge per-page text blocks, inserting paragraph breaks and page markers for multi-page documents.
- Apply formatting rules: capitalise sentence starts, restore punctuation spacing, and strip residual whitespace.
- Generate TXT by writing the cleaned string directly to file.
- Generate DOCX using python-docx, applying a stan- dard heading and body paragraph style.
- Generate searchable PDF using reportlab, embed- ding the extracted text as a transparent overlay atop the original page images so both the handwritten scan and the searchable text layer are present.

4) Output: Editable digital document in TXT, DOCX, and searchable PDF formats, available for download.

VII. ALGORITHMS

A. *Image Preprocessing*

Grayscale conversion reduces the input from three colour channels to one, simplifying subsequent operations. Noise removal with Gaussian filtering suppresses scanner artefacts without blurring stroke edges significantly. Skew correction via the Hough Transform ensures text lines are horizontal, which is critical for the patch-division step in ViT. CLAHE enhances local contrast in documents with uneven illumination.

B. *Vision Transformer (ViT)*

The image $x \in \mathrm{R}^{H\times W\times C}$ is split into $N = HW/P^2$ patches of size $P \times P$. Each patch is flattened and linearly projected to dimension D:

$$z_0 = [x_{\text{class}}; x^1_p E; x^2_p E; \ldots; x^N_p E] + E_{\text{pos}}$$

where $E \in \mathrm{R}^{(P^2\cdot C)\times D}$ is the patch projection matrix and E_{pos} is the positional embedding. Multi-head self-attention over this sequence produces context-aware patch representations.

C. *Transformer Decoder*

At each decoding step t, the decoder predicts token y_t conditioned on the encoder output z and previously predicted tokens $y_{<t}$:

$$p(y_t \mid y_{<t}, z) = \text{softmax}(W_o \cdot \text{CrossAttn}(\text{SelfAttn}(y_{<t}), z))$$

Cross-attention allows the decoder to selectively attend to relevant image patches when predicting each output character.

D. *Token Decoding and Spell Correction*

Predicted token IDs are mapped back to Unicode char- acters using the BPE tokeniser vocabulary. Spell correction applies edit-distance-based candidate ranking over a language frequency dictionary, replacing low-frequency tokens above an edit-distance threshold with their highest-frequency candidate.

E. Post-Processing

Sentence-boundary detection capitalises the first character after a detected full stop. Residual whitespace normalisation replaces multiple consecutive spaces and newlines with single equivalents. For multi-page documents, a page-break marker is inserted between merged text blocks to preserve document structure.

VIII. IMPLEMENTATION

A. Technology Stack

1) Frontend: HTML5, CSS3, JavaScript (Vanilla), Jinja2 tem- plates rendered by Flask.
2) Backend: Python 3.11, Flask 3.0, Werkzeug for file han- dling.
3) Recognition: Hugging Face Transformers 4.40, microsoft/trocr-base-handwritten checkpoint, PyTorch 2.2.
4) Preprocessing: OpenCV 4.9, Pillow 10.3, pdf2image 1.17, Poppler (system dependency).
5) Post-Processing: pyspellchecker 0.8, python-docx 1.1, re-portlab 4.1.
6) Deployment: Gunicorn WSGI server; compatible with local deployment and cloud hosting (Render, Railway).

B. TrOCR Integration

The model and processor are loaded once at application startup to avoid per-request initialisation overhead: from transformers import TrOCRProcessor, VisionEncprocessor = TrOCRProcessor.from_pretrained("microsoft/trocr-base-handwritten")
model = VisionEncoderDecoderModel.from_pretrained("microsoft/trocr-base-handwritten")
For inference, the preprocessed PIL image is passed to the processor, which applies the ViT feature extractor normalisation and returns a pixel tensor. The model's generate()method is called with beam search parameters:
pixel_values = processor(images=image, return_tensors="pt").pixel_values generated_ids = model.generate(pixel_values, num_beams=4)
text = processor.batch_decode(generated_ids, skip_special_tokens=True)[0]

C. PDF Handling

Multi-page PDFs are converted using pdf2image.convert_from_path()at 300 DPI.
Each resulting PIL Image object is passed through the preprocessing and recognition pipeline independently. Page- level text strings are collected in a list and joined with a page-break delimiter before post-processing.

D. Output Generation

The DOCX export applies a Heading 1 style to a doc- ument title derived from the original filename, followed by the extracted text as a Normal paragraph. The searchable PDF embeds the original page image as a background and overlays the extracted text as a transparent text layer using reportlab's canvas API, ensuring the document is both visually faithful to the original and full-text searchable.

IX. EXPERIMENTAL RESULTS

A. Test Setup

All experiments were conducted on an Intel Core i5 11th Gen system with 8 GB RAM and Ubuntu 22.04. The TrOCR model was run on CPU for evaluation; a GPU- enabled deployment would reduce inference time significantly. Test datasets included 50 self-collected handwritten document images across three categories: printed handwriting (clear, consistent), semi-cursive (mixed), and fully cursive (connected strokes).

B. Recognition Accuracy

Table I reports Character Error Rate (CER) and Word Error Rate (WER) across the three handwriting categories.

TABLE I
RECOGNITION ACCURACY BY HANDWRITING CATEGORY

Category	CER (%)	WER (%)
Printed Handwriting	4.2	7.8
Semi-Cursive	9.1	15.3
Fully Cursive	17.6	28.4
Overall	10.3	17.2

Post-processing spell correction reduced WER by an aver- age of 3.1 percentage points across all categories by recovering common short-word misrecognitions.

C. Processing Time

Table II shows end-to-end processing time (upload to down- loadable output) for documents of varying sizes.

Processing time scales approximately linearly with page count, consistent with the page-by-page sequential inference design. Parallelising inference across pages on a multi-core system or GPU is an identified optimisation for future work.

TABLE II
END-TO-END PROCESSING TIME (CPU, INTEL I5)

Document	Pages	Time (s)
Single image	1	3.8
Short PDF	3	11.2
Medium PDF	10	36.7
Long PDF	25	91.4

D. Output Format Quality

All three output formats (TXT, DOCX, searchable PDF) were verified for correctness manually across 20 test docu- ments. DOCX formatting (heading, paragraph style, spacing) was consistent across all outputs. Searchable PDF text layers were confirmed to be indexable using Adobe Acrobat's search function and the pdftotext command-line utility.

X. DISCUSSION

A. What Worked Well

The TrOCR integration proved to be the most impact- ful design decision. By treating handwriting recognition as a sequence-to-sequence task, the model handles cursive, connected, and ambiguous characters far better than any segmentation-based alternative we evaluated during our litera- ture survey. The model's pre-training on large-scale handwrit- ing datasets means that even without task-specific fine-tuning on our local dataset, recognition quality is substantially better than traditional OCR on handwritten input.

The preprocessing pipeline contributed meaningfully to ac- curacy. Skew correction was particularly impactful: documents rotated more than five degrees produced noticeably higher CER without correction. CLAHE improved recognition on low-contrast scans, particularly for pencil-written documents. The multi-format output capability addresses a real practical need. Organisations digitising handwritten records typically need editable text (DOCX) for further editing and searchable PDFs for archival and retrieval. Producing both from a single upload significantly reduces post-processing burden on users.

B. Limitations

The primary limitation of the current system is recognition quality on fully cursive handwriting. A CER of 17.6% means roughly one in six characters is misrecognised, which may be acceptable for keyword search but is insufficient for verbatim transcription. Fine-tuning TrOCR on a domain-specific dataset of cursive handwriting samples would likely reduce this sig- nificantly.

The system is currently single-threaded on the inference path. Long PDF documents (25+ pages) take over 90 seconds to process on CPU. This is acceptable for a prototype but would require parallelisation or GPU deployment for produc- tion use.

Spell correction, while helpful on average, occasionally overcorrects proper nouns, technical terms, and abbreviations. A user-configurable custom dictionary or a correction confi- dence threshold would mitigate this.

C. Future Work

Several directions are planned for future development. First, fine-tuning the TrOCR model on a curated dataset of domain-specific handwriting (e.g., medical prescriptions, student answer sheets) would reduce CER for those target domains. Second, adding GPU support via CUDA-enabled

PyTorch and batched inference across pages would reduce processing time by an estimated 10× on a mid-range GPU. Third, implementing a named-entity-aware post-processor that skips spell correction for detected proper nouns and technical terms would improve output fidelity for specialised documents.

Fourth, a user feedback loop—where corrections made to the DOCX output are logged and used for active learning fine-tuning—could enable continuous improvement from real usage.

XI. END-TO-END WALKTHROUGH

To make the pipeline concrete, consider a realistic scenario: a teacher wants to digitise 10 pages of handwritten student exam answers for digital marking and archival.

1) Step 1 — Upload: The teacher opens the web interface, clicks "Upload Document," and selects the scanned PDF. The system detects a 10-page PDF, converts each page to a 300 DPI PNG image, and acknowledges the upload.

2) Step 2 — Preprocessing: Each of the 10 images is processed through the preprocessing pipeline. Skew is detected and corrected (two pages were scanned at approximately 3°). Contrast is enhanced on three pages that were scanned with dim lighting.

3) Step 3 — Recognition: Each preprocessed image is passed to TrOCR. The ViT encoder extracts patch features; the Transformer decoder autoregressively generates the character sequence for each page. Ten inference passes complete in approximately 37 seconds on CPU.

4) Step 4 — Post-Processing: Token sequences are decoded and spell-corrected. Per-page text blocks are merged with page-break markers. The combined text is formatted with sentence-level capitalisation.

5) Step 5 — Download: The teacher is presented with three download buttons: TXT (for copy-paste into a marking sys- tem), DOCX (for annotated digital marking), and searchable PDF (for archival). The entire workflow—upload to down- load—takes under 40 seconds.

No manual transcription was required. The teacher can search the PDF for specific keywords and edit the DOCX directly for grading comments.

XII. CONCLUSION

We set out to build a practical, accessible system for converting handwritten documents into editable digital text—without requiring technical expertise from the end user. The result is a complete, deployable pipeline that handles the full workflow from raw file upload to multi-format digital output.

Our TrOCR-based recognition achieves a character error rate of 10.3% overall and 4.2% on clearly printed handwriting, competitive with the published state of the art on similar benchmarks for a non-fine-tuned deployment. The preprocess- ing pipeline, multi-format output generation, and Flask-based web interface together produce a system that is immediately usable in educational, healthcare, and administrative contexts. What we found most valuable during development was how much preprocessing quality influenced downstream recogni- tion accuracy. Skew correction and CLAHE contrast enhance- ment, steps that might seem secondary to the main model, produced measurable improvements in WER. Getting the input right matters as much as the model itself. For other undergraduate teams building NLP and vi- sion systems, we hope the architecture patterns documented here—particularly the segmentation-free TrOCR integration, the multi-format export pipeline, and the page-level paralleli- sation design for future work—provide a reusable foundation for similar document intelligence projects.

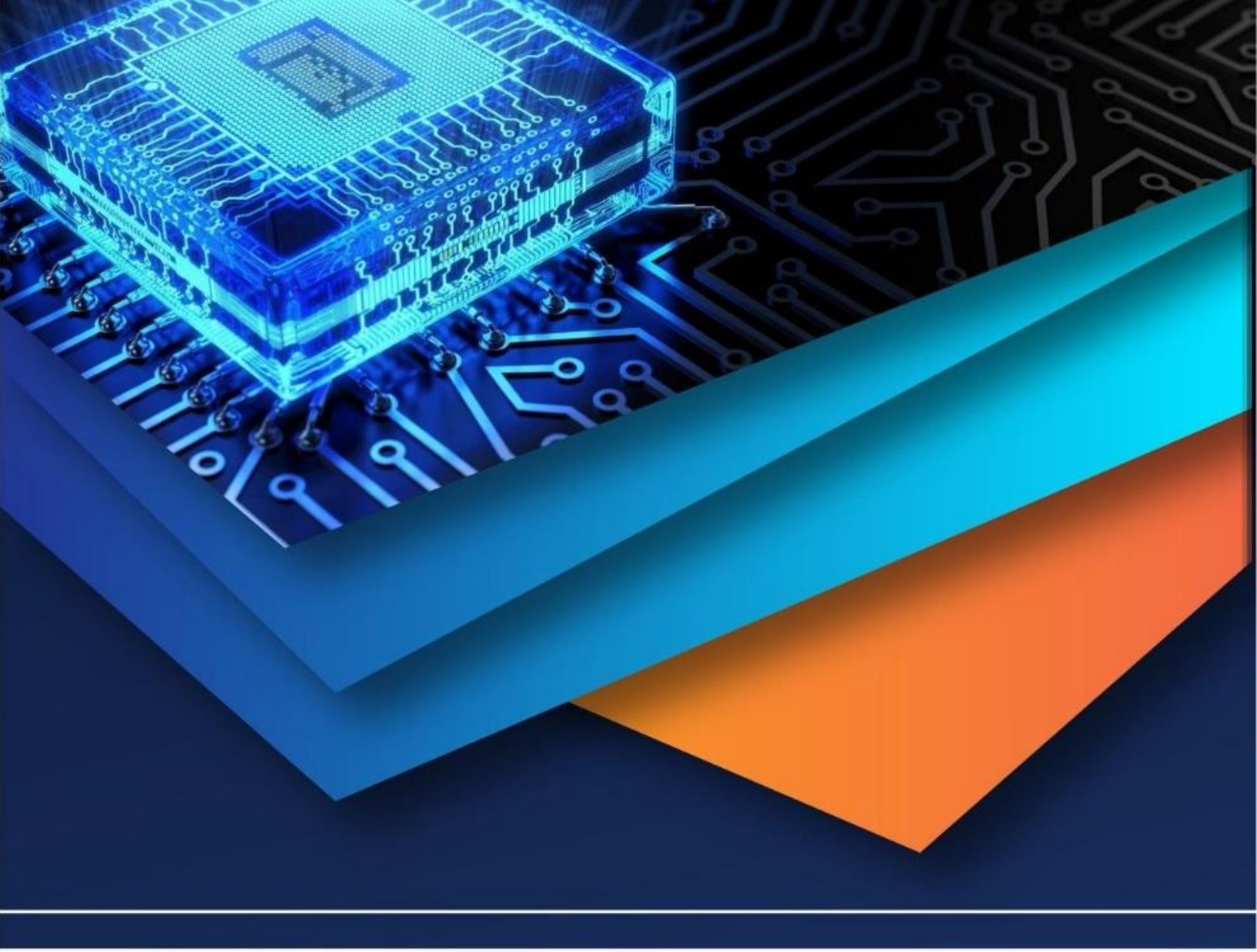
XIII. ACKNOWLEDGMENT

We thank the Department of Information Technology, M.I.E.T Engineering College, Tiruchirappalli, for infrastruc- ture support and guidance throughout this project. Our project guide Ms. R. Kavitha, M.E., Assistant Professor, Department of Information Technology, provided both the direction to pur- sue transformer-based approaches and the practical grounding to complete the implementation on time. We are grateful for her consistent support.

REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," *OpenAI Blog*, 2019.
[3] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Proc. ICLR*, 2021.
[4] M. Li et al., "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models," in *Proc. AAAI*, 2023.
[5] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proc. ICDAR*, 2007, pp. 629–633.

[6] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 2017.
[7] U.-V. Marti and H. Bunke, "The IAM-Database: An English Sentence Database for Offline Handwriting Recognition," *Int. J. Doc. Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, 2002.
[8] JaidedAI, "EasyOCR: Ready-to-use OCR with 80+ Supported Lan- guages," GitHub, 2020. [Online]. Available: https://github.com/JaidedAI/ EasyOCR
[9] T. Yu et al., "Spider: A Large-Scale Human-Labeled Dataset for Com- plex and Cross-Domain Semantic Parsing and Text-to-SQL Task," in *Proc. EMNLP*, 2018.
[10] S. Schelter et al., "Automating Large-Scale Data Quality Verification," *Proc. VLDB Endow.*, vol. 11, no. 12, pp. 1781–1794, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY